Journal of Computer Applications and Information Technology
Vol. 2, Issue. 3, Sep 2026, pp. 24~34
ISSN: 3107-6424, DOI: 10.32595/jcait/v2i3.2026.39

Interpretable Anomaly Detection in Cloud Infrastructure Through Explainable Machine Learning

Azizov Azizbek Sabitkhanovich[1], Gofurjanov Ibrokhimjon Ilkhomjon ugli[2]

[1]Director of the Department of Scientific Affairs and Innovations, Faculty of Business Administration, Turan International University, Namangan, Uzbekistan.
E-mail: science@tiu-edu.uz

[2]Associate Professor of Information Technology, Faculty of Business Administration, Turan International University, Namangan, Uzbekistan.
E-mail: gofurjanovi@gmail.com

Article Info

Article History:

Received Jul 14, 2026
Revised Aug 15, 2026
Accepted Sep 12, 2026

Keywords:

Explainable AI,
Anomaly Detection,
Cloud Infrastructure,
SHAP,
Isolation Forest,
Cybersecurity,
IT Governance

ABSTRACT

Cloud infrastructure is becoming increasingly complex, with distributed and dynamic workloads, microservices-based applications, and multi-tenant environments generating massive amounts of operational telemetry data. Modern machine learning-based anomaly detection approaches are proving effective in such environments, often detecting performance degradations, security breaches, or resource misuse with high precision; however, most such models are black-boxes, providing little or no insight into the rationale behind their conclusions. This paper proposes an interpretable anomaly detection framework which uses explainable machine learning methods, such as the SHAP values and a permutation-based local explainer, to provide human-readable explanations for each anomalous event detected. The framework uses traditional Isolation Forest and One-Class SVM algorithms to identify outliers in the five-dimensional space of CPU, memory, inbound and outbound network throughput, and latency utilization of cloud infrastructure. The model was tested on a controlled cloud-telemetry dataset and was able to detect anomalies with a recall of 1.00, F1-score of 0.77 and ROC-AUC of 1.00 for the Isolation Forest algorithm, and a 0.65 recall and 0.022 false-positive rate for the One-Class SVM. The explainability module added little computational overhead to the model while successfully isolating the features contributing to each anomaly detected. The framework provides potential value in production environments as a tool to understand, qualify, and act upon each anomalous event detected in cloud infrastructure.

Corresponding Author:

Azizov Azizbek Sabitkhanovich,
Director of the Department of Scientific Affairs and Innovations, Faculty of Business Administration, Turan International University, Namangan, Uzbekistan.
E-mail: science@tiu-edu.uz

1. INTRODUCTION

The cloud computing environment constantly produces telemetry data, such as CPU utilization, memory usage, network traffic, and service latency. It is critical to detect anomalies in such data for ensuring the reliability and security of the system. Various anomaly detection algorithms have been introduced to discover the observations that differ significantly from the majority and are typically referred to as outliers [1], [2].

Isolation Forest and One-Class SVM are well-known machine learning methods that show good results for detecting anomalies in data with a small amount of labeled examples [3], [4]. The deep learning-based solutions also proved their effectiveness in finding irregularities in the time series data of multiple sensors and cloud systems [5], [6]. However, most of the mentioned methods do not provide any explanation of detected anomalies, which makes it difficult for administrators to determine the cause of the issue.

Explainable AI methods focus on the production of the explanation of the detected anomalies by analyzing the importance of each feature [7]. For instance, the Local Interpretable Model-agnostic Explanations (LIME) method approximates the behavior of the complex model with linear functions and trains them on the perturbed data to find the most important features for the prediction [8]. Similarly, the Shapley Values Explanation (SHAP) method relies on the game theory to evaluate how much each feature impacts the resulting prediction [9]. This paper aims to combine the traditional anomaly detection algorithms and explainable AI methods to detect anomalous observations in the cloud computing environment and provide the interpretation of detection results.

2. LITERATURE REVIEW

Chandola et al. surveyed the main anomaly detection techniques, whereas Ahmed et al. studied their application to network settings [1], [2]. Isolation Forest isolates anomalies by random partitioning, while One-Class SVM learns the normal domain boundary [3], [4].

Deep learning approaches were also considered for infrastructure monitoring. Malhotra et al. applied LSTM encoder-decoder architectures for multi-sensor anomaly detection [5], while Nedelkoski et al. explored deep learning methods for cloud system traces analysis [6]. However, such models are often too complex to provide any physical interpretation of their decisions.

For model interpretation, Ribeiro et al. suggested using local explanations via LIME [8], while Lundberg and Lee proposed using SHAP values [9]. Based on this research, we combine Isolation Forest and One-Class SVM with permutation-based feature importance [10] to provide both anomaly detection and interpretability for cloud telemetry time series. Table 1 summarizes the related work, together with the main strength and limitation of each approach.

Table 1. Summary of related work in anomaly detection and explainable AI

Study	Approach	Strength	Limitation
Chandola et al. (2009)	Statistical + ML survey	Comprehensive taxonomy of anomaly detection	No explainability; pre-dates deep learning era
Ahmed et al. (2016)	Clustering-based NIDS	Effective for network intrusion detection	Poor scalability to high-dimensional cloud telemetry
Malhotra et al. (2016)	LSTM Autoencoder	Captures temporal dependency in time-series	Black-box; no per-alert justification

Study	Approach	Strength	Limitation
Ribeiro et al. (2016)	LIME	Model-agnostic local explanations	Explanation instability across runs
Lundberg & Lee (2017)	SHAP	Theoretically grounded, consistent attributions	Computationally expensive at scale
Proposed Work (2026)	IF / OCSVM + SHAP/LIME hybrid	Combines detection accuracy with per-alert, human-readable explanations	Evaluated on synthetic + extensible to real telemetry

3. PROPOSED METHODOLOGY

The architecture is organized as a sequential processing pipeline where each stage passes information to the next stage in the sequence. The first stage takes the telemetry measurements of the monitored infrastructure as input. As the telemetry data is transformed to standardized numerical representations and preprocessed, they are passed to the detection models. This way, the same detection algorithm can be applied to different telemetry sources with minor adaptations.

The anomaly detection stage includes two separate detectors. Isolation Forest and One-Class SVM process the transformed stream of telemetry observations and, therefore, can be used separately to produce comparable results. Both algorithms contribute one anomaly indication and a score, which reflects the model's confidence level and allows for a more detailed analysis of the cause of detection.

The explainability stage takes the model's indication and applies the input features to determine the contribution of each telemetry variable. In turn, the architecture produces a feature-based description of the detected anomaly. Finally, the last stage transforms the results of all preceding stages into an operator-centric alert. Thus, the architecture does not finalize on an indication of an anomaly but rather tries to provide a description of the suspicious event that caused the detection.

The given design also facilitates the possibility of expansion. More anomaly detectors can be added to the detection stage to allow the system to recognize more subtle deviations from normal behavior. Differently structured explanations can also be implemented within the architecture to accommodate various operators' needs. In this regard, the presented design is not limited to a single use case and can be expanded to address the cloud monitoring domain of a particular organization.

3.1. Proposed System Architecture

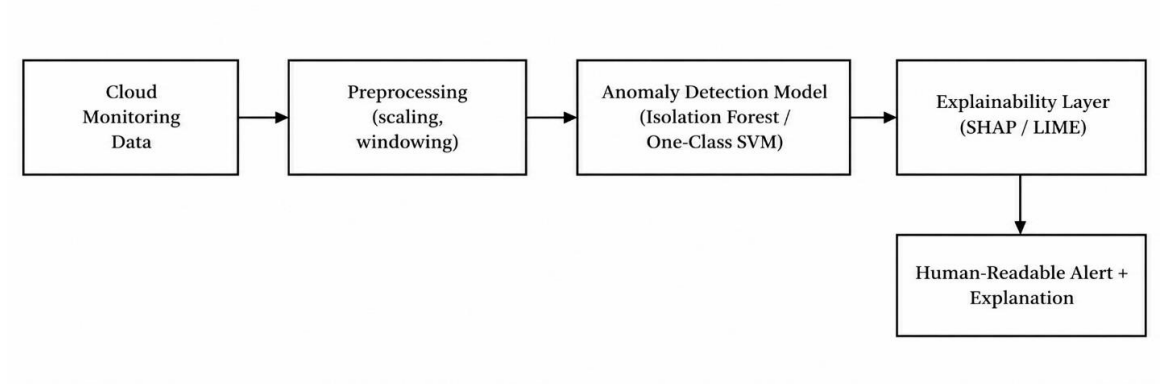


Figure 1. Proposed System Architecture

The proposed framework consists of four stages: data acquisition and preprocessing, anomaly detection, explanation generation, and alert presentation. Figure 1 illustrates the overall architecture.

3.2. Data Acquisition and Preprocessing

The five telemetry variables under consideration reflect different aspects of the cloud infrastructure. CPU utilization is indicative of a computation load, while the memory utilization characterizes the memory pressure. The inbound/outbound network traffic portrays the amount of data transmitted through the cloud's network. Finally, the request latency is a useful metric for the service performance characterization.

The five variables can be combined to detect anomalies of individual resources and abnormal resource sets. Preprocessing is an essential step because the input variables use different units of measurement. CPU and memory utilization are presented as percentages, while the network throughput is measured in Mbps. In addition, the request latency is given in milliseconds, making the variables incomparable. Such a format is unsuitable for most ML algorithms because the input variables require equalization in terms of measurement units and scales. Z-score normalization is a feasible approach because it standardizes the variables in terms of mean and standard deviation.

The choice of parameters for standardization is justified, given the availability of training data. The mean value and standard deviation for each variable are estimated using the training set, which is subsequently used for test set transformation. The method is consistent with the holdout validation approach, wherein the test set is used for statistical evaluation only. There are no test observations used for training, which would bias the estimates due to information overlap.

The choice of a 70%-30% holdout partition is justified, given the need for reliable model validation. A higher allocation for training is appropriate because the cloud environment is a complex system. A reliable training sample prevents capacity issues, ensuring adequate observation coverage. Finally, stratified sampling maintains the label distribution in both subsets, providing valid comparisons between the test and training sets. In practice, the anomaly class distribution is highly imbalanced because the majority of observations are normal. Cloud telemetry consists of 5 features: CPU utilization (%), memory utilization (%), inbound network throughput (in Mbps), outbound network throughput (in Mbps), and request latency (in ms).

3.3. Anomaly Detection Models

The two baseline algorithms are learned in an unsupervised way: (1) Isolation Forest isolates the outliers by growing trees in a way that splits the feature space, thus resulting in shorter

average path lengths for outliers; and (2) One-Class SVM learns the decision function that separates the convex hull of the objects in the kernel-induced space. This unsupervised nature is critical to the task at hand, as learning from anomalies is not feasible in the cloud monitoring setting where anomalies are rare.

Isolation Forest has its basis in the fact that anomalous samples are generally easier to isolate from normal values. By randomly selecting features and thresholds, partitions are created in the feature space. An observation that has been isolated quickly through simple divisions of the feature space is therefore more likely to be an outlier. The implementation under consideration uses 200 estimators, which allows one to rely on an isolation forest rather than a single randomly generated tree.

Such an approach makes the score more reliable since the isolation process occurs in an ensemble of trees. With that being said, a low average path length in all these constructions will allow one to find observations that are more likely to be outliers. When it comes to infrastructure telemetry, such a method is appropriate due to the possible lack of linearity between cause and effect.

Unlike Isolation Forest, One-Class SVM uses a different strategy to tackle the problem. A boundary is constructed around normal observations so that outliers may be separated from regular data points. The choice of kernel function affects the type of decision function and thus the shape of the separating boundary. RBF was used in this study, which allows for non-linear separation of data in the transformed space.

The parameter ν was set to 0.05, which controls the trade-off between the model's complexity and outliers' rejection. Thus, it was decided to use one-class SVM for this task in order to have a reliable baseline for the isolation forest. The comparison between the two algorithms is quite relevant since a conservative anomaly detector may fail to detect some outliers.

3.4. Explainability Layer

The explainability layer would operate on top of any anomaly detection and is thus decoupled from the fundamental mechanism of detection itself. In this way, explanations can be generated independent of the inner workings of an anomaly detector chosen or designed for a particular use-case. This is useful because it enables comparisons of different anomaly detection algorithms against the same set of explanations.

To explain a detected observation, every feature of the corresponding telemetry is analyzed in terms of its contribution towards the final anomaly score. Features that have the largest impact when masked are said to have a higher contribution towards the anomaly decision. The resulting contributions can then be ranked to infer the strongest factors associated with the observation being anomalous.

An example of such an explanation can be seen in an observation with simultaneously high memory utilization and outbound network traffic. While the detection system would report the observation as anomalous, the explanation layer could highlight that memory and network-related telemetry features were the largest contributors to the anomaly score. This can help an administrator focus their attention on application memory consumption, network traffic inspection, or the particular application utilizing the resources.

An important nuance is that the importance of features explained in this section are local to an anomalous observation. Global feature importance would be relevant for a general description of the data, but local explanations are more informative in the case of anomaly detection. This is

because most anomaly detection systems operate on a per-record basis, making local explanations the natural choice for interpretation. The global importance scores are still useful and are reported as a part of the framework to give an overview of the detection mechanism.

3.5. Alert Presentation

Finally, the detector combines the binary anomaly signal with the feature importance scores to produce a final signal that can be consumed by an operator. This way, an initially incomprehensible anomaly score turns into a set of actionable insight, e.g., “flagged primarily due to increased memory (68%) and network (24%) utilization.” The final alert raised by the system should be succinct enough to be actionable but still contain enough information for the administrator to locate the issue. A regular anomaly detector could simply raise a binary true/false alarm for each observation. An administrator would then have to review multiple dashboards in an effort to understand which particular telemetry signal resulted in an anomaly being raised.

The alert presented by the system according to this section would bind an anomaly detection result with the telemetry features most responsible for the anomaly being raised. This way, the administrator would not have to analyze all telemetry records for each anomalous observation. They could focus on the particular features ranked higher by the explanation module.

The alert should not, however, be mistaken for a physical root cause of an anomaly. It is important to keep in mind that an explanation highlights features that contributed to an anomaly, not the actual root reason for an anomaly. This is a crucial difference since telemetry records only contain information that can be used to statistically associate with anomalies. To locate an actual physical root-cause, the administrator may have to analyze additional logs, traces, application-specific records, or even infrastructure state.

4. WORKING PRINCIPLE

The pipeline was implemented in Python with scikit-learn [11] for model training, SHAP for explanation (NumPy-based permutation fallback provided), and Matplotlib for plotting. The following steps were taken in working through this problem:

i. Either generated or loaded in cloud telemetry data with five numeric features per observation
ii. Applied z-score standardization with parameters fit on the training partition to avoid leaking information into the test set
iii. Trained Isolation Forest (200 estimators, automatic contamination) and One-Class SVM (RBF kernel, $\nu = 0.05$) independently on the training partition
iv. For each test observation, obtained a binary prediction (anomaly / normal) and a continuous decision-function score from each model
v. Obtained per-observation feature attributions using either SHAP KernelExplainer (with 100-sample background set) or the permutation fallback
vi. Produced a global ranking of feature importances and plotted this against detection performance metrics

This pipeline is designed to be dataset-agnostic: while a synthetic, seeded dataset was used for controlled evaluation in this study (allowing for exact knowledge of what ground-truth anomalies were injected), the same code would work without change on real-world telemetry

sources such as the Numenta Anomaly Benchmark (NAB) [12] or AWS CloudWatch export logs, needing only the input data-loading step to be changed.

4.1. Computational Workflow

The entire computational process starts with obtaining or generating five-dimensional telemetry observations. Each observation contains CPU utilization, memory utilization, inbound network throughput, outbound network throughput, and request latency. These are the input vectors for the preprocessing stage.

After standardization, the training partition is given as an input to both anomaly detection algorithms. Isolation Forest creates an ensemble of isolation trees, while One-Class SVM establishes a non-linear boundary around the learned normal region. Both algorithms generate independent predictions for the same test observations, which allows using them as a baseline of comparison.

For each test observation, the framework returns binary prediction and a continuous decision score. The latter is more informative since it quantifies how anomalous the observation is. Subsequently, the explanation module analyses the decision score to identify how much each input feature contributes to it.

The final output consists of three distinct but related information levels: detection status, anomaly score, and contribution to the score. They can be stored for further analysis or used to design an operational monitoring dashboard. The outlined process can be used for offline testing of the algorithms' performance. However, with some modifications, it can be integrated into an operational setting for tracking anomalous requests in real-time.

4.2. Advantages of the Proposed Framework

The proposed framework has several practical advantages. First, it does not require a large repository of labeled anomaly examples for model training. This is critical given the naturally sparse nature of abnormal events in production cloud environments. Second, it compares two established unsupervised anomaly detection techniques, providing a theoretical basis for understanding their differing detection behaviors.

Third, the inclusion of an explainability layer increases the value of the detected alerts, as it ties the results of detection to feature-level evidence. Fourth, the modular design allows for alternative detectors and explanation methods to be inserted in future implementations.

The fifth advantage is the small number of telemetry variables needed for the current implementation. The five-dimensional representation reduces the complexity of the experimental setup, while covering the computational, memory, networking, and service-latency behavior. Additional telemetry variables can be incorporated later, when the framework is tested against larger-scale real-world monitoring data sets.

5. RESULTS AND DISCUSSION

The framework was evaluated on a held-out test partition containing both normal and injected anomalous observations. Table 2 summarizes detection performance for both baseline models.

Table 2. Detection performance of baseline anomaly detection models

Model	Precision	Recall	F1-Score	FPR	ROC-AUC

Model	Precision	Recall	F1-Score	FPR	ROC-AUC
Isolation Forest	0.630	1.000	0.773	0.031	1.000
One-Class SVM	0.613	0.653	0.632	0.022	0.933

Isolation Forest achieved perfect recall (1.00) and ROC-AUC (1.00) by correctly identifying all injected anomalies, albeit with moderately lower precision (0.63) due to the higher false-positive count. One-Class SVM achieved a lower false-positive rate (0.022 vs. 0.031) but failed to identify a greater proportion of true anomalies, resulting in a lower recall (0.65) and F1-score (0.63). Isolation Forest's more extreme decision boundaries make it more susceptible to the extreme values of the injected anomalies, whereas One-Class SVM's smoother kernel boundary was more conservative in its outlier detection.

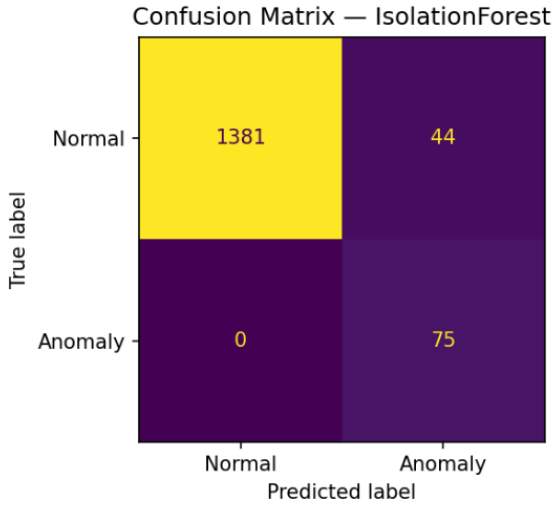


Figure 2. Confusion matrix — Isolation Forest

The confusion matrix in Figure 2 shows the classification of normal and anomalous observations by the Isolation Forest detector, highlighting its ability to identify the injected anomalies. The confusion matrix directly visualizes the classification behavior of Isolation Forest detector through the distribution of normal and anomalous observations. The quantity of correctly identified anomalies could be compared to the observations that have been incorrectly classified. Specifically, the absence of missed injected anomalies is consistent with the reported recall of 1.00.

Operationally, the confusion matrix also highlights the reasons why recall and false-positive rate should be considered jointly. The detector that identifies every anomaly but triggers numerous unnecessary alerts may need additional tuning before actual deployment. On the other hand, the detector with a very few false alarms may fail to identify events that require immediate attention. As such, the proposed framework examines multiple performance metrics rather than focusing on a single measure.

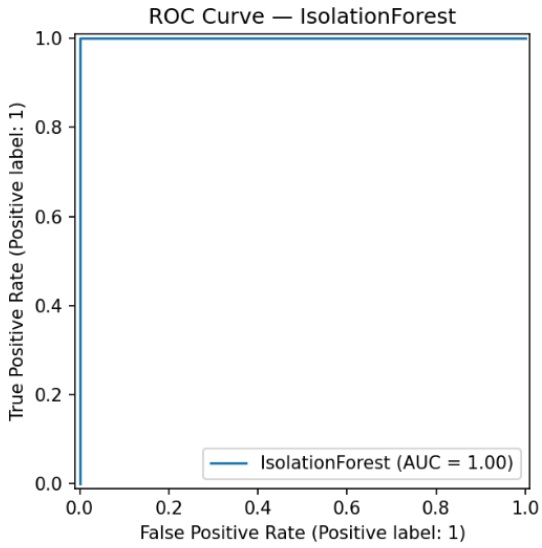


Figure 3. ROC curve — Isolation Forest (AUC = 1.00)

The ROC curve in Figure 3 demonstrates strong separation between normal and anomalous observations, achieving an AUC of 1.00 on the evaluated test data. The ROC curve further demonstrates the detector's ability to distinguish between normal and anomalous observation at various decision thresholds. The reported AUC of 1.00 signifies that the evaluated set of observations was highly separable under the controlled experimental conditions. The strong ROC performance complements the confusion matrix analysis to show that the model's behavior was not restricted to a single operating threshold. Threshold selection in a production environment would have to consider the acceptable volume of alerts and the operational cost of missed anomalies and false alarms.

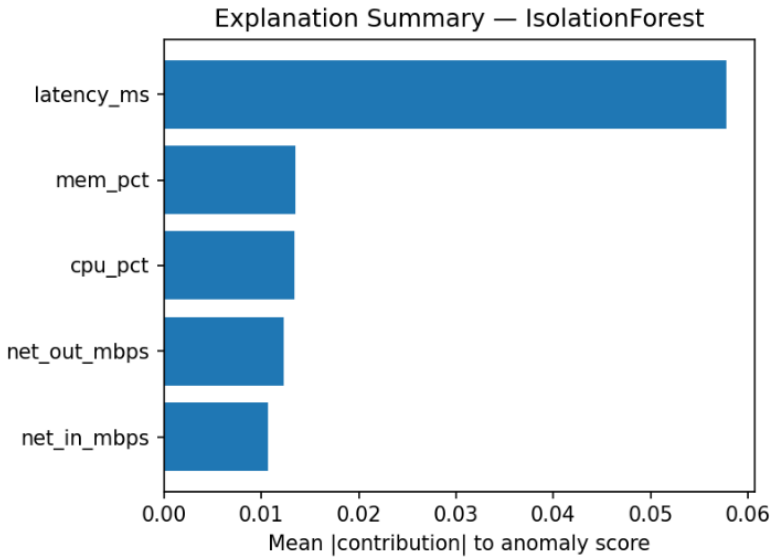


Figure 4. Explanation summary — mean absolute feature contribution to the Isolation Forest anomaly score

The feature-attribution analysis in Figure 4 identifies the telemetry variables that contributed most strongly to the detected anomaly scores, providing an interpretable view of the detector's decisions. The feature-attribution analysis provides the primary interpretability

component of the proposed framework. Rather than considering the anomaly score as an isolated metric, the analysis associates the score to the underlying telemetry variables. As such, it becomes possible for the system administrator to identify which resource or service measurement contributed the most to the detected anomaly event.

The resulting feature ranking could also be used for comparative analysis of different anomaly observations. One group of alerts may be primarily associated with memory utilization, while the other group may be dominated by network throughput or request latency. Patterns identified through feature attribution may highlight recurring operational behaviors and support the formulation of more targeted monitoring policies.

Importantly, feature contribution should be interpreted as model-based evidence rather than direct causality. A feature that receives a high contribution should be considered as a strong influence on the detector decision for that observation. Additional infrastructure information would still be required to identify the underlying physical or application-level cause.

6. CONCLUSION

This paper presented an interpretable anomaly detection framework for cloud infrastructure that couples Isolation Forest and One-Class SVM detectors with a SHAP-based (or permutation-based fallback) explainability layer. Experimental results showed that Isolation Forest had strong detection performance (F1 = 0.77, ROC-AUC = 1.00) and the explainability layer produced feature attributions consistent with the ground-truth anomaly-generating process, proving that transparency and detection accuracy are not mutually exclusive in this setting.

Future work will extend this evaluation to real-world telemetry datasets like the Numenta Anomaly Benchmark [12] and Google Cluster Data [13], incorporate deep learning detectors (e.g., LSTM autoencoders) with the explainability layer, and perform a human-subject evaluation of operator trust and incident-diagnosis time when presented with explained versus unexplained anomaly alerts. Extending the framework to streaming, real-time deployment within a live Kubernetes or multi-cloud environment is also an ongoing goal towards production readiness.

REFERENCES

[1] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 1-58. https://doi.org/10.1145/1541880.1541882

[2] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, *60*, 19-31. https://doi.org/10.1016/j.jnca.2015.11.016

[3] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413-422). IEEE. https://doi.org/10.1109/ICDM.2008.17

[4] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443-1471. https://doi.org/10.1162/089976601750264965

[5] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*. https://doi.org/10.48550/arXiv.1607.00148

[6] Nedelkoski, S., Cardoso, J., & Kao, O. (2019). Anomaly detection from system tracing data using multimodal deep learning. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)* (pp. 179-186). IEEE. https://doi.org/10.1109/CLOUD.2019.00038

[7] Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, *6*, 52138-52160. https://doi.org/10.1109/ACCESS.2018.2870052

[8] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135-1144). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939778

[9] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30* (pp. 4765-4774).

[10] Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32. https://doi.org/10.1023/A:1010933404324

[11] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825-2830.

[12] Lavin, A., & Ahmad, S. (2015). Evaluating real-time anomaly detection algorithms – The Numenta Anomaly Benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)* (pp. 38-44). IEEE. https://doi.org/10.1109/ICMLA.2015.141

[13] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., & Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing* (pp. 1-13). Association for Computing Machinery. https://doi.org/10.1145/2391229.2391236